МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”

(ФГБОУ ВО «ВГУ»)

Факультет компьютерных *наук*

Кафедра информационных систем

Разработка веб-приложения «Пушкограм»

Курсовой проект по дисциплине «Технологии программирования»

*09.03.02 Информационные системы и технологии*

*Информационные системы и сетевые технологии*

*6 семестр 2021/2022 учебного года*

Обучающийся\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Будаев А.В.

Обучающийся\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Быканов Н.С.

Обучающийся\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Шалиткин В.А.

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Тарасов В.С.

Воронеж 2022

**Содержание**

[Введение](#_heading=h.gjdgxs) 3

1. [Анализ предметной области](#_heading=h.30j0zll) 4

1.1 [Глоссарий](#_heading=h.3znysh7) 4

1.2 [Технический обзор](#_heading=h.2et92p0) 5

1.3 [Сравнение аналогов](#_heading=h.tyjcwt) 5

2. [Постановка задачи](#_heading=h.vx1227) 7

2.1 [Требования к функциональной части](#_heading=h.3fwokq0) 7

2.2 [Технические требования](#_heading=h.1v1yuxt) 7

2.3 [Требования к интерфейсу](#_heading=h.4f1mdlm) 8

3. [Схема базы данных](#_heading=h.2u6wntf) 21

4. [Диаграмма вариантов использования](#_heading=h.19c6y18) 22

5. [Диаграмма состояний](#_heading=h.3tbugp1) 22

6. [Диаграмма последовательностей](#_heading=h.28h4qwu) 23

7. [Реализация](#_heading=h.nmf14n) 25

8. [Тестирование](#_heading=h.1mrcu09) 28

9. [Заключение](#_heading=h.46r0co2) 28

[Список литературы](#_heading=h.2lwamvv) 30

# Введение

Социальные сети в наше время занимают важное место в жизни современного общества. От обычного общения, до рекламы и политики, все аспекты жизни так или иначе пустили корни в социальные сети.

Однако у социальных сетей есть две большие и пугающие проблемы, которые «‎Пушкограм»‎ стремится избежать. Первая - это сконцентрированность основной части данных в интернете на серверах всего нескольких компаний, таких как Google и Amazon. Это означает, что данные пользователей фактически не принадлежат пользователям и в любой момент компания может их использовать для своих нужд или попросту удалить. Это также, как правило, закреплено пользовательским соглашением.

Второй проблемой является неизбежная цензура и политизация. Интернет уже давно перестал быть «‎диким западом» социального взаимодействия. Крупные компании и государства научились контролировать ‎взаимодействие в интернете, вводить блокировки и цензуру.

Обе этих проблемы решают децентрализованные вычисления и хранение данных. С помощью библиотеки GunJS возможно создать любое количество узловых серверов, связывающих клиенты и синхронизирующих данные между собой. Таким образом невозможно повлиять на данные извне или удалить их, а сервера заблокировать навсегда, так как всегда можно запустить новый узловой сервер. Также такая система защищена от «‎‎атаки 51%», характерной для децентрализованных систем при правильной архитектуре приложения.

Для достижения поставленной цели необходимо решить следующие задачи:

* Разработать структуру базы данных;
* Разработать структуру приложения;
* Разработать документацию;
* Произвести тестирование разработанного приложения.

# Анализ предметной области

Целью является создание web-приложения и связанного с ним telegram-бота для публикации и оценки фотографий. Разработанное программное обеспечение может быть использовано, как замена социальной сети Instagram, сильно подверженной цензуре и печально известной использованием данных пользователей неэтичными способами.

## Глоссарий

API - набор программных модулей и протоколов, позволяющих Серверу Приложению обмениваться данными.

Авторизация - предоставление Пользователю прав на использование ресурсов Приложения в соответствии с его статусом и на основе данных, полученных при Регистрации.

База данных системы - внутренняя база данных системы.

Диаграмма вариантов использования (use case diagram) — модель анализа с указанием действующих лиц, которые могут взаимодействовать с системой для выполнения задач, и различные варианты использования, в которых может участвовать действующее лицо.

Неавторизованный пользователь (гость) - человек, который может авторизоваться в мобильном приложении.

Пользователь (user) — клиент, который взаимодействует с системой непосредственно или косвенно (например, пользуется результатами работы системы, хотя не генерирует эти результаты). Также называется конечным пользователем.

Узловой сервер – сервер, указанный в клиенте, как пир, и предназначенный только для хранения данных и синхронизации клиентов между собой.

Функциональное требование — описание поведения системы в определенных условиях.

## Технический обзор

В отличие от аналогов, указанных ниже, «‎Пушкограм»‎ использует модель децентрализованного хранилища данных и произведения основных вычислений на клиенте, в силу их легковесности. Благодаря этому узловой сервер служит лишь для синхронизации данных на всех клиентах, подключенных к нему.

Всё это позволяет создать максимально быстрое и легковесное приложение, а разработчикам нужно лишь взаимодействовать с Front-end частью, так как на сервере не могут проводиться какие либо вычисления, иначе теряется смысл децентрализации.

## Сравнение аналогов

В качестве аналогов можно рассмотреть такие сервисы, как Twitter и Instagram.

Twitter - американский сервис микроблогов и социальная сеть для публикации коротких сообщений.

К преимуществам можно отнести:

* Возможность широкой кастомизации профиля пользователя.
* Возможность быстрого создания публикаций и взаимодействия с ними других пользователей.
* Система рекомендаций.

Минусы:

* Невозможность создавать посты длиной более 280 символов.
* Серьёзная и не всегда оправданная цензура, как со стороны администрации, так и со стороны пользователей.
* Отсутствие защиты от государственного воздействия.

Instagram - американская социальная сеть, основной функцией которой является возможность публикации фотографий и взаимодействие пользователей путём комментариев или личных сообщений.

Плюсы:

* Возможность создавать временные посты - «Истории»
* Лента неперематываемых видео, аналогичная приложению TikTok
* Система рекомендаций, основанная сложных алгоритмах вычисления интересов пользователей.

Минусы:

* Случаи использования данных пользователей в форме, не указанной в пользовательском соглашении.
* Полная централизация.
* Серьёзная цензура.
* Тяжеловесность сервиса на клиенте.

# Постановка задачи

Разработать приложение для работы с базой данных, предназначенной для хранения и изменения информации о:

* Фотографиях;
* Текстовых записях (постах);
* Оценках пользователей (лайках);
* Данных клубов читателей.

## Требования к функциональной части

К основным функциональным требованиям можно отнести:

* Производить авторизацию пользователей.
* Возможность публиковать публикации.
* Возможность публиковать текстовые публикаций с помощью telegram-бота.
* Возможность просмотра 5 крайних публикаций при помощи telegram-бота.
* Возможность оценки публикации.
* Возможность редактирования и форматирования текста публикации.

## Технические требования

Приложение должно обеспечивать:

* Авторизацию его пользователей (читателей, администраторов) посредством электронной почты и пароля, которые хранятся и идентифицируются внешней (по отношению к приложению) системой авторизации;
* При удалении книги автоматически должна удаляться информация со всех таблиц, связанных с удаляемой книгой.

## Требования к интерфейсу

Пользовательский интерфейс должен:

* Обеспечивать интуитивно понятное взаимодействие пользователя с системой.
* Сократить время загрузки страницы и обеспечить возможность доступа ко всей логике приложения в пределах одной страницы.
* Не давать пользователю возможность создания публикации, не соответствующей формату, предусмотренному в системе.
* Обеспечить быструю авторизацию или регистрацию пользователя.

Макеты интерфейса:

После запуска web-приложения, пользователь переходит на стартовую страницу авторизации. (рис. 1), где ему будет предложено войти в приложение. В силу философии поддержки относительной анонимности, используемой в приложении, вход в профиль совмещён с регистрацией нового пользователя, в случае если любое из полей отличается от существующих в системе.

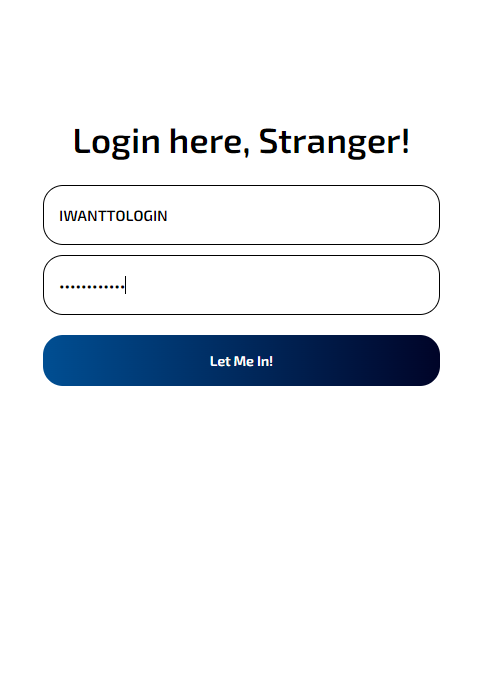


Рисунок 1 – Страница входа в приложение.

Нажав кнопку «‎‎Let Me In!» пользователь попадает на основную страницу сайта, где он может увидеть публикации других пользователей, отсортированные в порядке даты создания (рис. 2).

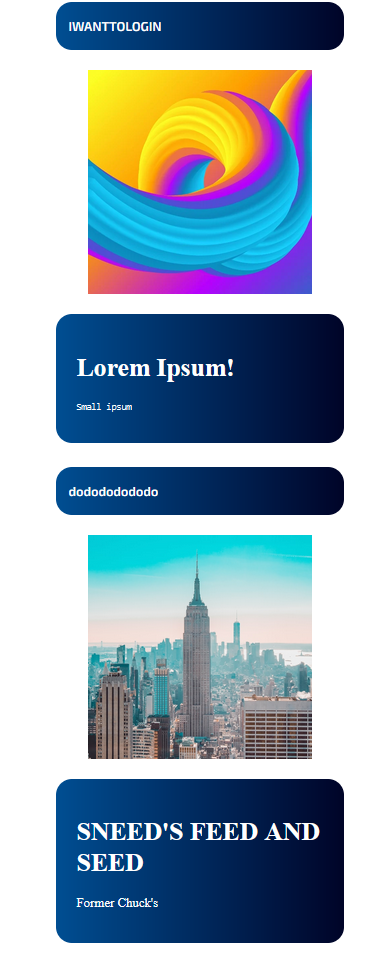


Рисунок 2 – Лента публикаций.

Над публикацией указано имя пользователя, а под публикацией – текстовая часть, в формате RichText.

Справа от ленты находится форма создания публикаций (Рис. 3). По-умолчанию кнопка отправки публикации заблокирована и будет разблокирована лишь после загрузки файла и создания текстовой части публикации.

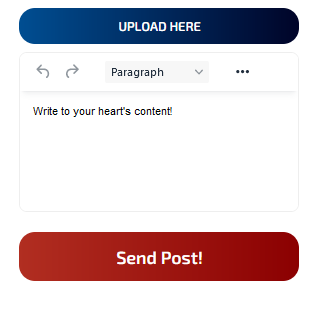


Рисунок 3 – форма создания публикации.

После заполнения необходимых полей, кнопка отправки публикации меняет цвет на зелёный и отправка разблокируется (Рис. 4).

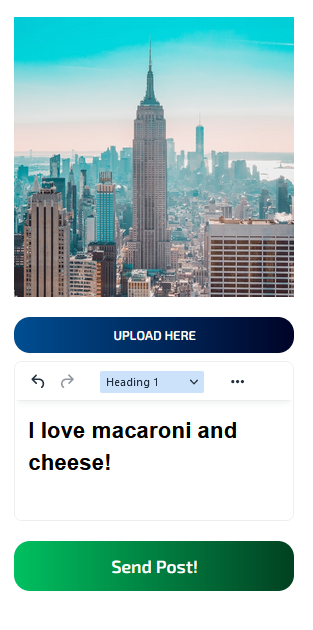


Рисунок 4 – Публикация, допустимая к отправке.

Вне зависимости от разрешения фотографии, соотношение её сторон будет приведено к 1:1.

Редактор текста предоставляет возможность базового форматирования текста, выравнивания, создания списков, изменения, сдвигов текста, выделения и указания стиля (Рис. 5, Рис.6).

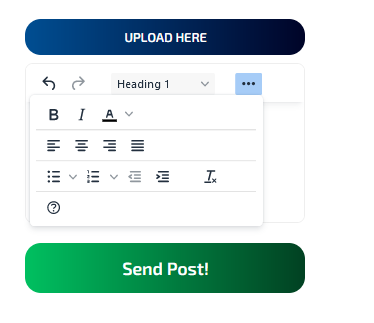


Рисунок 5 – Окно редактирования форматирования текста.

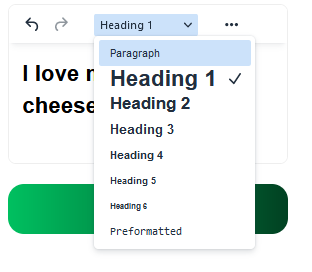


Рисунок 6 – Окно редактирования стиля выделенного текста.

При наведении на публикацию будет проиграна анимация появления кнопок «Like» и загрузки фотографии из публикации (Рис. 7).

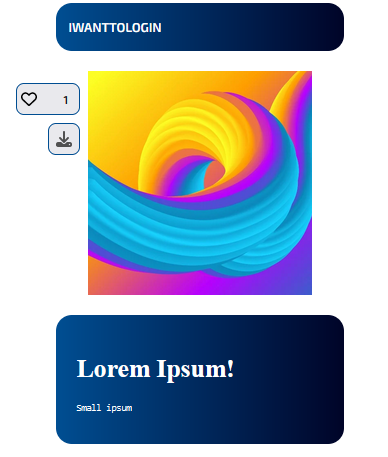
‎

Рисунок 7 – Демонстрация публикации с кнопками оценки и загрузки изображения публикации.

Цифра на первой кнопке соответствует количеству оценок, оставленных другими пользователями.

При входе в Telegram-бот пользователя приветствует главный экран с кнопкой “Начать” (Рис. 8).

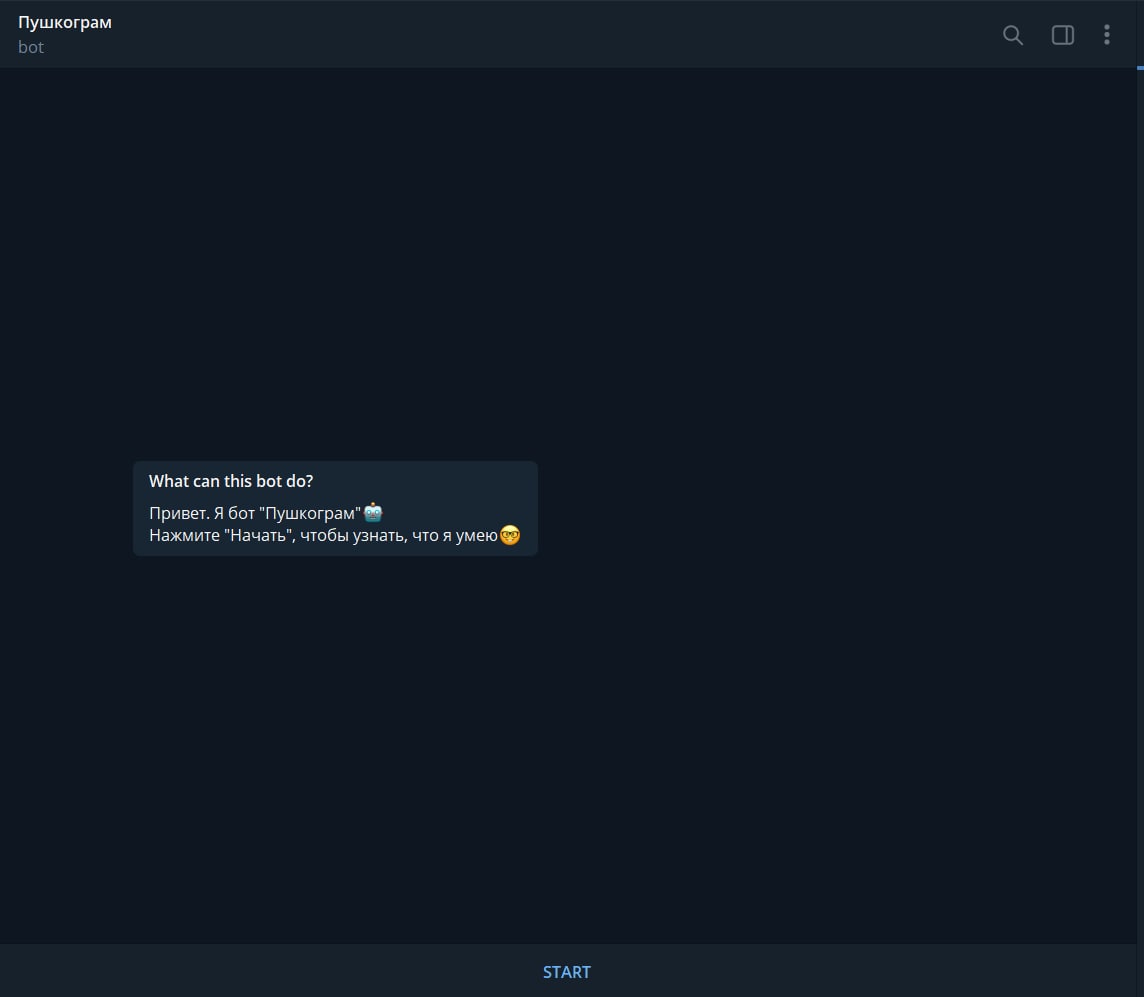
‎

Рисунок 8 – Демонстрация стартовой страницы Telegram-бота.

При нажатии на основную иконку бота можно увидеть описание, и контакты разработчиков для связи на случай технических проблем. (Рис. 9)

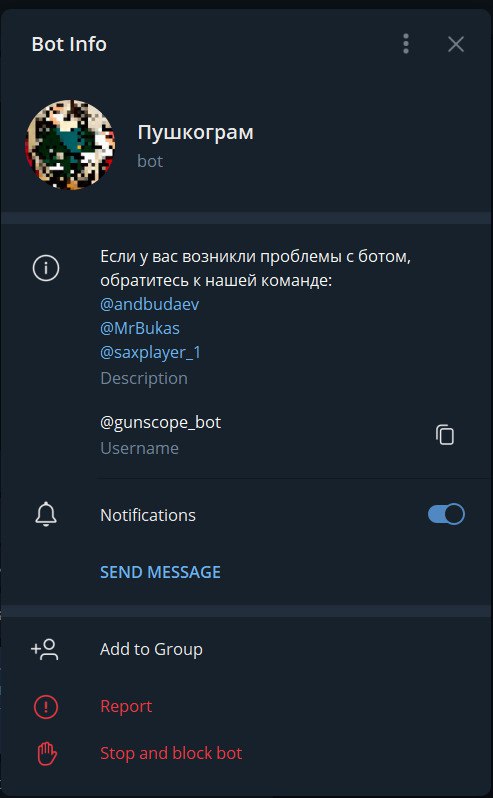
‎

Рисунок 9 – Демонстрация описания Telegram-бота.

Затем, после нажатия кнопки “Начать” (или “Start” на английской версии программы), пользователь видит меню управления (Рис. 10).

По умолчанию ему доступны кнопки “Войти в аккаунт” и “Смотреть посты”. Возможность просматривать посты доступна для всех пользователей, даже не сохранивших свои данные для входа (Рис. 11).

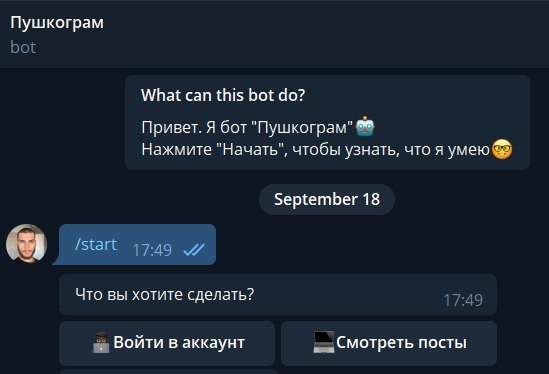
‎

Рисунок 10 – Меню управления для неавторизованного пользователя.

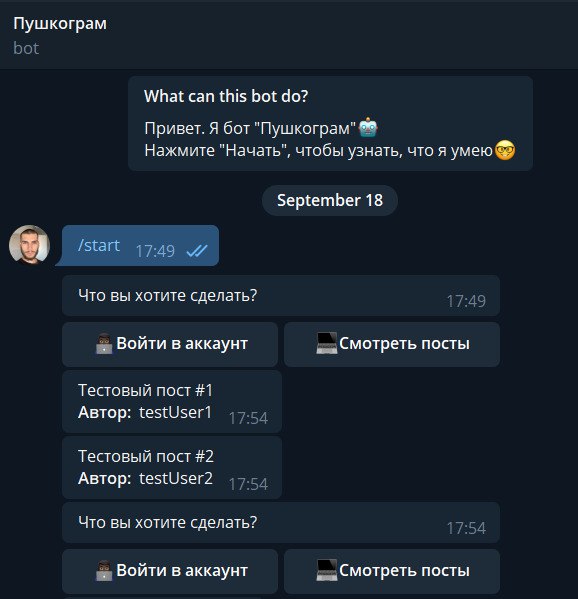
‎

Рисунок 11 – Демонстрация работы кнопки “Смотреть посты”.

Однако, чтобы отправлять посты, пользователю необходимо войти в аккаунт. Для этого после нажатия кнопки “Войти в аккаунт” последовательно отправляются сообщения для заполнения сперва логина, а затем пароля. (Рис. 12) Причем каждое из этих действий можно отменить, нажав кнопку “Отмена”, и вернуться в главное меню.

‎

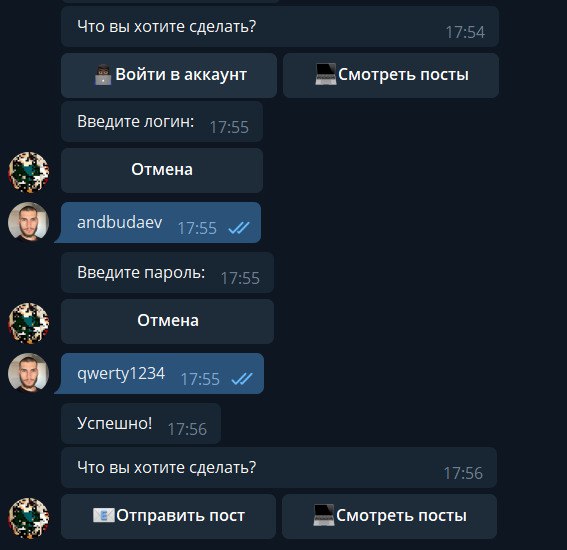
‎

Рисунок 12 – Ввод логина и пароля.

После успешного ввода логина и пароля, появляется доступ к кнопке “Отправить пост”. Эта кнопка активирует появление сообщения с вводом текста поста (Рис. 13). Это действие также можно отменить кнопкой “Отмена”.

После успешной отправки поста, снова появляется меню для навигации.

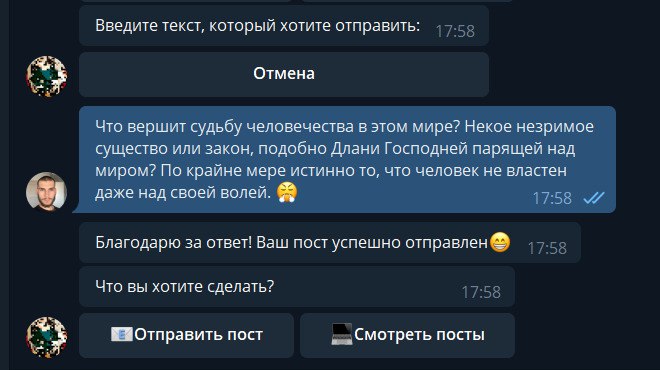
‎

Рисунок 13 – Демонстрация отправки поста.

# Схема базы данных

Схема базы данных представлена на рисунке 14:

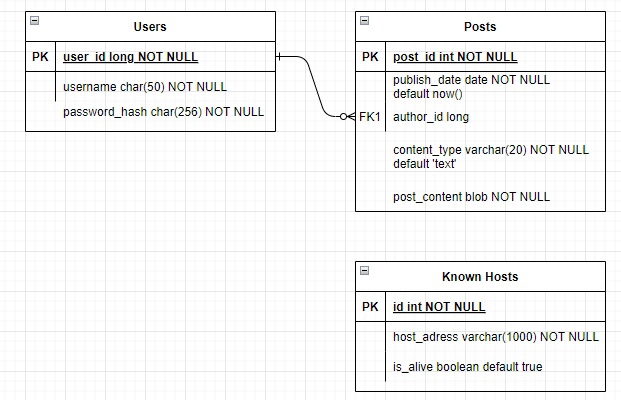


Рисунок 14 – Схема базы данных

Users – таблица, содержащая информацию о данных пользователей, зарегистрированных в приложении.

Posts– таблица, содержащая все созданные пользователями посты и информацию о них.

Known Hosts– таблица, содержащая адреса других узловых серверов.

# Диаграмма вариантов использования

На рисунке 15 представлена диаграмма вариантов использования:

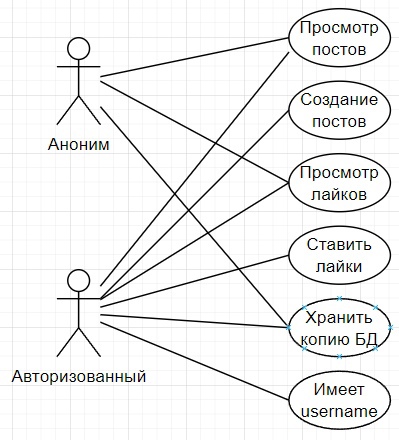


Рисунок 15 – Use case диаграмма

# Диаграмма состояний

На рисунке 16 представлена диаграмма состояний:

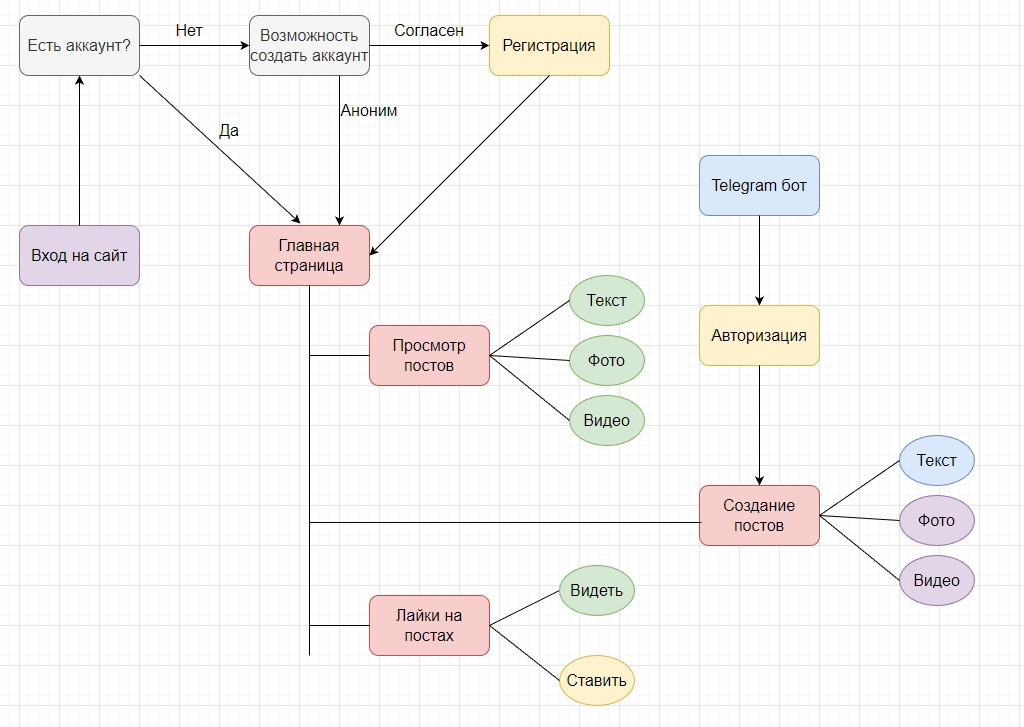


Рисунок 16 – Диаграмма состояний

# Диаграмма последовательностей

Диаграмма последовательностей для незарегестрированного пользователя изображена на рисунке 17:

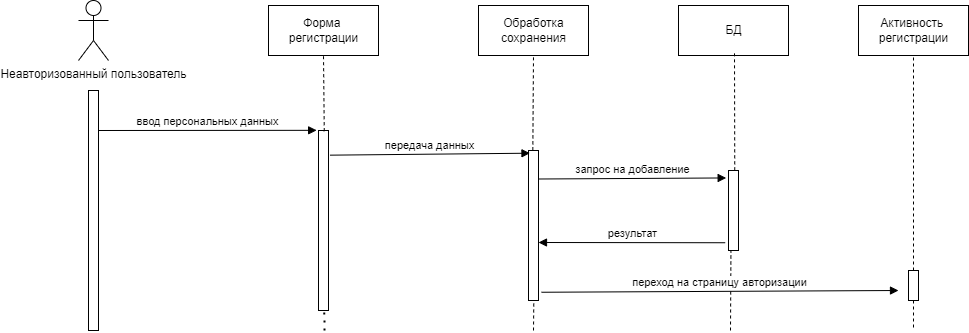


Рисунок 17 - Диаграмма последовательностей для незарегестрированного пользователя

Диаграмма последовательностей для авторизованного пользователя представлена на рисунке 18:

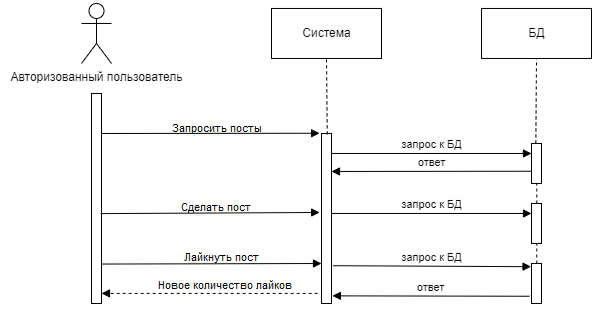


Рисунок 18 - Диаграмма последовательностей для авторизованного пользователя

# Реализация

Front-end:

Клиентская часть представляет собой React SPA web-приложение, выполняющее всю логику на клиенте.

В папке components хранятся простые компоненты, из которых состоит приложение.

В папке pages находятся собранные из компонентов страницы, с которыми может взаимодействовать пользователь.

Хранение данных и авторизация производятся с помощью функционала GUN.API и Redux.

Файл App.js в папке Pages отвечает также за создание экземляра класса GUN и задание списка узловых серверов, с которыми необходимо синхронизироваться при попытке получить или сохранить какие либо данные.

Вся концепция децентрализованного хранения данных заключается в синхронизации локального хранилища данных браузера с хранилищем узловых серверов, указанных, как пиры.

Back-end:

Фактически состоит из файла server.js, содержащего запуск web-сервера на необходимом порте, импорт необходимых библиотек и декларацию сервера, как узлового в приложении, что позволяет ему при каждом запросе данных клиентом синхронизировать данные с клиентом. При этом синхронизируются не все данные, а лишь те, к которым клиент получал доступ, что позволяет значительно сократить вычислительную нагрузку на сервер.

Файл Endpoints содержит пути для взаимодействия telegram-бота с сервисом путём HTTP запросов.

Файл Service реализует внутреннюю логику, позволяющую боту взаимодействовать с хранилищем и отправлять посты на web-клиент.

Telegram-bot:

Клиентская часть реализована благодаря возможностям мессенджера Telegram. MTProto API (он же Telegram API) — это API, через который приложение Telegram связывается с сервером.

Для его работы задействован промежуточный сервер с HTTPS-интерфейсом, который шифрует трафик и обеспечивает связь с Telegram API.

В нашем случае мы используем вебхук на <https://andbots.ru/bot/webhook>. Куда приходят обновления от Telegram, которые мы обрабатываем, а затем уже отправляем ответ в зависимости от результата.

Back-End реализован при помощи возможностей PHP-фреймворка Yii2 Advanced, но в рамках этой задачи была изменена структура фреймворка, с целью отказаться от разделения на Front-End и Back-end (предусмотренной фреймворком по умолчанию), и был оставлен только Back-end.

Так как фреймворк Yii2 построен на структуре MVC, основной точкой входа проекта является файл /backend/controllers/BotController.php. В нем и находится основной экшн - actionWebhook(), на который приходят все обновления от Telegram.

После получения очередного обновления, мы оперируем моделью /backend/models/BotUser.php, хранящую все данные о текущем пользователе (уникальным ключем является его ID в Базе Данных Telegram). BotUser является представлением единственной таблицы в БД бота (Рис. 14). В эту таблицу мы сохраняем все необходимые для функционирования бота данные.

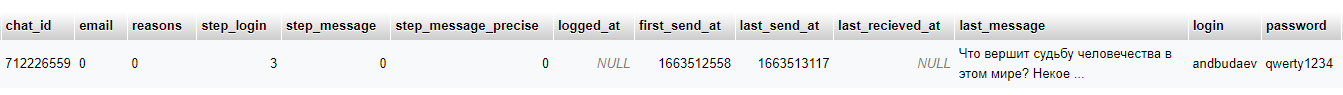


Рисунок 14 – Таблица bot\_user

В папке /backend/helpers/ находятся вспомогательные классы (валидация, работа со строками, и т.д.).

В папке /backend/bots/Bot.php находится самописный класс, реализующий отправку HTTP запросов согласно документации Telegram Bot API.

Для отправки постов из Telegram, после обработки события отправляется HTTP-запрос на основной Back-end, хранящий данные о пользователе (username, password), и получающий в ответ результат запроса.

Для получения постов с основного сайта используется метод отправки пустого HTTP-запроса на основной Back-end, и получающего в ответе необходимые данные. Эти данные обрабатываются и отправляются уже Telegram-сообщением пользователю, запросившему посты.

# Тестирование

В ходе тестирования необходимо проверить методы для взаимодействия с базой данных (соответствие реального результата ожидаемому).

Для функционального тестирования необходимо вручную проверить все варианты использования системы различными группами пользователей.

1. Функциональное тестирование:

* установка и запуск приложения;
* авторизация (при вводе верной комбинации имени пользователя и пароля открывается главное меню с возможными вариантами действий; в случае неверных данных – появляется сообщение об ошибке);
* выход из системы (при нажатии на кнопку выхода из системы сессия пользователя уничтожается, открывается главная страница);
* написание поста.

2. Тестирование безопасности:

* данные пользователя (пароли) не передаются в открытом виде;
* в полях, с вводом пароля и подтверждением пароля, данные скрываются астерисками.

3. Тестирование удобства использования

* Корректное отображение элементов на устройствах с различными разрешениями экранов;
* Все тексты правильно выровнены;
* Все тексты без орфографических и грамматических ошибок;
* Корректный возврат на предыдущий экран.

# Заключение

В ходе данной работы было разработано web-приложение поддерживаемое всеми современными браузерами, позволяющее публиковать фотографии и оценивать их.

Также были проанализированы существующие решения, и оценены их основные преимущества и недостатки, был проведён анализ предметной области и выбраны средства реализации и обхода недостатков существующих решений.

В следующих версиях приложения в качестве улучшений можно добавить больше возможностей для взаимодействия пользователей, такие как личные сообщения и комментарии.

# Список литературы

1. Вигерс Карл Разработка требований к программному обеспечению/ Вигерс Карл, Битти Джой - Пер. с англ. — М. : Издательство «Русская редакция» ; СПб. : БХВ-Петербург, 2014. — 736 стр.